

Rainbow: Documentation for the MAPLE package

David Joyner

3-3-96

1 Introduction

The **rainbow** MAPLE package is really a game derived from MasterBall, a Rubik's cube-like puzzle. This game was initially designed as an educational instrument to motivate students to learn some MAPLE commands as well as to introduce some more advanced students to some related problems in group theory (e.g., the word problem) and theoretical computer science (e.g., efficient methods of solving the Rubik's cube). Some of this documentation assumes a little familiarity with group theory.

Some rules for MasterBall: A masterball sphere has 32 tiles of 8 distinct colors. We shall assume that the masterball is in a fixed position in space. A geodesic path from the north pole to the south pole is called a **longitudinal line** and a closed geodesic path parallel to the equator is called a **latitudinal line**. There are 8 longitudinal lines and 3 latitudinal lines. In spherical coordinates, the longitudinal lines are at $\theta = n\pi/4$, $n = 1, \dots, 8$, and the latitudinal lines are at $\phi = \pi/4, \pi/2, 3\pi/4$. The sphere shall be oriented by the right-hand-rule - the thumb of the right hand wrapping along the polar axis points towards the north pole. We assume that one of the longitudinal lines has been fixed once and for all. This fixed line shall be labeled "1", the next line (with respect to the orientation above) as "2", and so on.

Allowed manuevers: One may rotate the masterball east-to-west by multiples of $\pi/4$ along each of the 4 latitudinal bands or by multiples of π along each of the 8 longitudinal lines. The set of all possible finite sequences of such manuevers forms a group, called the **masterball group**. There are

$4 + 8 = 12$ generators of this group (this set is not minimal). The group will be described a little more later.

A **facet** will be one of the 32 subdivisions of the masterball created by these geodesics. A facet shall be regarded as immobile positions on the sphere and labeled either by an integer $i \in \{1, \dots, 32\}$ or by a pair $(i, j) \in [1, 4] \times [1, 8]$, whichever is more convenient at the time. If a facet has either the north pole or the south pole as a vertex then we call it a **small** facet. Otherwise, we call a facet **large**. A **coloring** of the masterball will be a labeling of each facet by one of the 8 colors in such a way that

- (a) each of the 8 colors occurs exactly twice in the set of the 16 small facets,
- (b) each of the 8 colors occurs exactly twice in the set of the 16 large facets.

A **move** of the masterball will be a change in the coloring of the masterball associated to a sequence of manuevers as described above.

If we now identify each of the 8 colors with an integer in $1, \dots, 8$ and identify the collection of facets of the masterball with a 4×8 array of integers in this range. To “solve” an array one must, by an appropriate sequence of moves corresponding to the above described rotations of the masterball, put this array into a “rainbow” position so that the matrix entries of each column has the same number. Thus the array

$$\begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array}$$

is “solved” and corresponds to the picture below.

The array

6	7	8	1	2	3	4	5
1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8

corresponds to the picture below.

2 Notation

The generators for the latitudinal rotations are denoted r_1, r_2, r_3, r_4 , so for example,

$$r_1 : \begin{array}{cccccccccc} 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \\ 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 \\ 31 & 32 & 33 & 34 & 35 & 36 & 37 & 38 \\ 41 & 42 & 43 & 44 & 45 & 46 & 47 & 48 \end{array} \xrightarrow{\quad} \begin{array}{cccccccccc} 12 & 13 & 14 & 15 & 16 & 17 & 18 & 11 \\ 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 \\ 31 & 32 & 33 & 34 & 35 & 36 & 37 & 38 \\ 41 & 42 & 43 & 44 & 45 & 46 & 47 & 48 \end{array} .$$

As you look down at the ball from the north pole, this move rotates the ball *clockwise*. The other moves r_2, r_3, r_4 rotate the associated band of the ball in the same direction - clockwise as viewed from the north pole. The generators

for the longitudinal rotations are denoted f_1, f_2, \dots, f_8 , so for example,

$$f_1 : \begin{array}{ccccccccccccc} 12 & 13 & 14 & 15 & 16 & 17 & 18 & 11 & & 44 & 43 & 42 & 41 & 16 & 17 & 18 & 11 \\ 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & \longmapsto & 34 & 33 & 32 & 31 & 25 & 26 & 27 & 28 \\ 31 & 32 & 33 & 34 & 35 & 36 & 37 & 38 & & 24 & 23 & 22 & 21 & 35 & 36 & 37 & 38 \\ 41 & 42 & 43 & 44 & 45 & 46 & 47 & 48 & & 15 & 14 & 13 & 12 & 45 & 46 & 47 & 48 \end{array}.$$

With these rules, one can check the relation

$$f_5 = r_1^4 * r_2^4 * r_3^4 * r_4^4 * f_1 * r_1^4 * r_2^4 * r_3^4 * r_4^4.$$

There are similar identities for f_6, f_7, f_8 . Also, one can check that

$$r_1 = (f_3 * f_7)^{-1} * r_4^{-1} * f_3 * f_7.$$

There are similar identities for r_2, r_3, r_4 .

Identifying the facets of the masterball with the entries of the array

$$\begin{array}{cccccccc} 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 16 & 15 & 14 & 13 & 12 & 11 & 10 & 9 \\ 24 & 23 & 22 & 21 & 20 & 19 & 18 & 17 \\ 32 & 31 & 30 & 29 & 28 & 27 & 26 & 25 \end{array},$$

we may express the generators of the masterball group in disjoint cycle notation as a subgroup of S_{32} (the symmetric group on 32 letters):

$$\begin{aligned} r_1^{-1} &= (1, 2, 3, 4, 5, 6, 7, 8), \\ r_2^{-1} &= (9, 10, 11, 12, 13, 14, 15, 16), \\ r_3^{-1} &= (17, 18, 19, 20, 21, 22, 23, 24), \\ r_4^{-1} &= (25, 26, 27, 28, 29, 30, 31, 32), \\ f_1 &= (5, 32)(6, 31)(7, 30)(8, 29)(13, 24)(14, 23)(15, 22)(16, 21), \\ f_2 &= (4, 31)(5, 30)(6, 29)(7, 28)(12, 23)(13, 22)(14, 21)(15, 20), \\ f_3 &= (3, 30)(4, 29)(5, 28)(6, 27)(11, 22)(12, 21)(13, 20)(14, 19), \\ f_4 &= (2, 29)(3, 28)(4, 27)(5, 26)(10, 21)(11, 22)(12, 23)(13, 24), \\ f_5 &= (1, 28)(2, 27)(3, 26)(4, 25)(9, 20)(10, 19)(11, 18)(12, 17), \\ f_6 &= (8, 27)(1, 26)(2, 25)(3, 32)(16, 19)(9, 18)(10, 17)(11, 24). \\ f_7 &= (7, 26)(8, 25)(1, 32)(2, 31)(15, 18)(16, 17)(9, 24)(10, 23), \\ f_8 &= (6, 25)(7, 32)(8, 31)(1, 30)(14, 17)(15, 24)(16, 23)(9, 22), \end{aligned}$$

These generate a subgroup G (denoted `perm_ball` in `rainbow`) of S_{32} of order $N = 437763136697395052544000000$.

The group G acts on the set of all possible colorings. The masterball itself contains several symmetries which prevent this group from acting freely. For example, the red "north pole facet" can be switched with the red "south pole facet" without changing the masterball's position. We may also rotate the entire ball "east-to-west" by $\pi/4$ radians without affecting the masterball's position. As another example, we can simply turn the ball over, swapping the north pole with the south pole. There are a total of $n_1 * n_2 * n_3 * n_4 = 2^{20}$ such symmetries, where n_1 denotes the number of north-south pole swaps (2), n_2 denotes the number of latitudinal rotations by multiples of $\pi/4$ (8), n_3 denotes the number of like-colored "north-south pole facet" swaps (2^8), and n_4 denotes the number of like-colored "upper middle-lower middle facet" swaps (2^8). The number of positions of the masterball is therefore $N/2^{20} \cong 4.1 \times 10^{20}$. This is only about 10 times larger than the number of positions of the 3×3 Rubik's cube.

In fact, the elements $f1, f2, f3, f4, r1, r2, r3, r4$ generate this group. (Exercise: Convince yourself of this.) In fact, according to GAP's `AbStab.g` share package, the move f_1 is not necessary either, so G is generated by at most 7 elements.

This group is about ten million times larger than the group of the Rubik's cube (see, for example, §2.4 in [B]). The derived subgroup of this group is of index 4 and after that the decreasing chain of derived subgroups stabilizes, according to MAPLE calculations.

3 MAPLE commands

First, open a MAPLEV4 worksheet and load the packages `linalg` and `plots`. If you want to use the `master_squares` or `square_draw` commands, also load the `geom` package. If you want to use any of the group theory commands also load the `group` package.

Copy the `rainbow.txt` file into a directory (eg, `c:\maplev4\share\games\rainbow` if you are using a dos machine) and read it into your maple session (by typing
`read('c:/maplev4/share/games/rainbow/rainbow.txt');`
for example).

3.1 To play the game

I shall now assume that you have a MasterBall in front of you. The initial masterball position will be described as a 4×8 matrix as above. Call this matrix $A0$. Now pick a word in the group, say $w := [r_1, f_1, r_3^3]$; and type $rb(w, A0)$; to "rotate" the ball (hence the "rb") according to the group element w . You try to choose manuevers which eventually will put the sphere into "rainbow" form.

3.2 The solution strategy

Step 1: The idea is to first get all the middle bands aligned first, so you get ball corresponding to a matrix of the form

$$\begin{array}{cccccccc} * & * & * & * & * & * & * & * \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ * & * & * & * & * & * & * & * \end{array}$$

Here, * denotes any color. We have labeled the colors on the masterball as 1, 2, ..., 8 in order of occurrence.

We describe a method, which I call "crab fishing", for achieving this. (Mathematically, this amounts to performing some carefully choosen commutators.) Without too much trouble you can always assume that we have one column aligned. You may need to flip or rotate the ball a little bit to do this. Call this aligned column "column 1" and call the color in column 1, "color 1". We want to get the middle two entries in column 2 aligned. Call the color in the (2, 3)-entry "color 2".

We want to get color 2 in the (2, 2)-entry. The remaining large color 2 tile is the "crab" we will "fish" for. Hold the ball in front of you in such a way that column 2 is slightly to the left of center and column 3 is slightly to the right of center. There are 4 facets in the right upper middle band, 4 facets in the left upper middle band, 4 facets in the right lower middle band and 4 facets in the left lower middle band. A flip about the center on the right half (i.e., perform f_2) exchanges these. We may assume that color 2 is on one of the four facets in the right lower middle band. (If it isn't you need to apply f_2 first). Now perform $r_2^{-1}f_2^{-1}r_2f_2$: first perform r_2^{-1} (this is "baiting the hook"), then f_2^{-1} ("putting the hook in the water"), then r_2 ("setting

the hook”), and finally f_2 (“reeling in the hook”). You may or may not have color 2 in the (2, 2) place like you want but the color 1 stripe is intact. If necessary, try again. After at most 4 tries you’ll be successful.

Step 2: Repeat this “crab fishing” strategy to get color 2 in the (1, 2) position (using $r_1^{-1}f_2^{-1}r_1f_2$ in place of $r_2^{-1}f_2^{-1}r_2f_2$). Now, by turning the ball over if necessary, repeat this idea to get color 2 in the (4, 2) position. Now you have two ”aligned” stripes on your ball - color 1 in column 1 and color 2 in column 2. We say, in this case, that columns 1 and 2 have been ”solved”.

Step 3: Repeat this for columns 3 and 4.

Step 4: Assume that we have four columns completely aligned: columns 1, 2, 3 and 4. Place the ball so that columns 1, 2 are slightly to the left of center and columns 3, 4 are slightly to the right. Perform a flip about the middle (i.e., an f_3). Now rotate the entire ball by 45° so that two solid stripes are “left-most” (say columns 1,2) and two solid stripes are “right-most” (say columns 3,4). The ball looks more symmetric about the center this way. The flip about the middle will be simply f_4 in this set up. Now try using the following types of manuevers: (“middle switching”): $f_4, f_8, r_2f_4r_2^{-1}f_4^{-1}r_2^{-1}f_4r_2f_4^{-1}, r_3f_4r_3^{-1}f_4^{-1}r_3^{-1}f_4r_3f_4^{-1}, r_2f_8r_2^{-1}f_8^{-1}r_2^{-1}f_8r_2f_8^{-1}, r_3f_8r_3^{-1}f_8^{-1}r_3^{-1}f_8r_3f_8^{-1}$. These may be performed in any order. The point is that after each such manuever, the 4 solid columns will remain solid, whereas the “mixed up” columns may remain mixed up. My (limited) experience has been that this creates, eventually, aligned middle bands for all 8 columns.

Step 5: “So close, but yet so far”. After step 4, we have at most 4 north pole facets out of alignment and at most 4 north pole facets out of alignment. A similar strategy as in step 4 for the north pole and the south pole (i.e., replace r_2, r_3 in the ”middle switching” manuevers of step 4 by r_1, r_2 , resp.) does not usually lead to a solved ball. Call this new collection of manuevers ”polar switching”. ”Polar switching” will lead to some better situations: my experience has been that ”polar switching” creates, eventually, 5 or 6 aligned columns and the 2 or 3 remaining columns have only the south pole facet out of alignment.

Step 6a: Assume that we have 5 aligned columns (say, columns 1,2,3,4,8) and the 3 remaining columns (5, 6, 7) have only the south pole facet out of alignment. In this case, we need a cyclic permutation of order three to ”solve” the ball. There is a manuever of 86 moves which does this. (This was discovered using GAP’s `AbStab.g` and checked both by hand and by computer.) It is

```

r4^(-2)*f3*r4*f4*r4*f4*r4^(-1)*
r1^2*f3*r4^(-1)*f3*r1^(-3)*f3*r4^(-1)*
f3*r1*f3*r4^2*f3*r1*f3*r4^(-1)*
f3*r1^(-1)*f3*r4^2*f3*r1*
f3*r4^(-1)*f3*r1^(-1)*f3*r4^(-4)*f3*r1*
f3*r4*f3*r1^(-1)*f3*r4*f3*
r1^(-2)*f3*r4^(-1)*f3*r1*f3*r4^(-1)*f3*r1^(-1)*
f3*r4^2*f3*r1^4*f3*r4*f3*
r1^(-3)*f3*r4^(-1)*f3*r1*f3*r4^2*
f3*r1^(-1)*f3*r4*f3*r1^3*f3*r4*
f3*r1^(-3)*f3*r4^(-1)*f3*
r1^2*f3*r4^(-1)*f3*r1^(-1)*f3

```

Step 6b: Assume that we have 6 aligned columns and each of the 2 remaining columns have only the south pole facet out of alignment. In this case, we need a cyclic permutation of order two to “solve” the ball. Unfortunately, the only maneuver I know of to do this (discovered using GAP’s `AbStab.g`) is too long to be practical so it will not be given here (it is, however, given in the appendix in MAPLE notation as `long2cycle`).

3.3 Some other useful moves

In this subsection, we use list notation for the moves, so they may be directly plugged into the rainbow “rb” command if desired.

Some 2-cycles on middle facets:

The following move acts as the permutation (9,17) on the masterball:

```

NSmid_swap:=[r3,r2,f1,r2,f1,r3,f1,r2^(-1),f1,r3^(-2),
f1,r2,f1,r3,f1,r2^(-1),f1,r3^4,f1,
r2,f1,r3,f1,r2^(-1),f1,r3^(-2),f1,r2,f1,
r2^(-1),r3^2,r2,f1,r2^(-1),f1,r3^2,f1,r2,
f1,r3^(-1),f1,r2^(-1),f1,r3^(-1),f1,r2,f1,
r3,f1,r2^(-1),f1,r2,r3^(-2),f1,r2^(-1),f1,
r3^(-1),f1,r2,f1,r2^(-1),r3,f1,r2,f1,
r2^(-1),f1,r2,f1,r2^(-1),r3^2,r2,f1,
r2^(-1),f1,r2,f1,r2^(-1),f1,r2,r3^(-1),
f1,r2^(-1),f1,r3,f1,r2,f1,r2^(-1),r3^2,r2^(-1)];

```

The following move acts as the permutation (9,10) on the masterball:

```

EW_swap:=[f4^-1,f3,f1^-1,r2,f1^-1,r1,r4,r3,
r2,f1^-1,r2,f1,r3^(-1),f1^-1,r2^(-1),f1^-1,r3^(-1),f1^-1,
r2,f1^-1,r3^(-1),f1^-1,r2^(-1),f1^-1,r2^(-1),r3^3,r2,f1^-1,
r2,f1^-1,r3^(-1),f1^-1,r2^(-1),f1^-1,r3^2,f1^-1,r2,
f1^-1,r3^(-1),f1^-1,r2^(-1),f1^-1,r2^(-1),r3^2,r2^2,f1^-1,
r2^(-1),f1^-1,r2,f2^-1,r4^(-1),r1^(-1),f1^-1,r3^(-1),f1^-1,r2^(-1),f1^-1,
r3^(-1),f1^-1,r2,f1^-1,r3,f1^-1,r2^(-1),f1^-1,r3^2,
f1^-1,r2,f1^-1,r3^(-1),f1^-1,r2^(-1),f1^-1,r3^(-1),f1^-1,r2,
f1^-1,r3,f1^-1,r2^(-1),f1^-1,r2,f1^-1,r2^(-1),f1^-1,
r3^2,r2^2,r3^(-1),f1^-1,r2,f1,r3^2,f1^-1,r2^(-1),
f1^-1,r2^(-1),r3^(-3),r2,f1^-1,r2,f1^-1,r3^2,f1^-1,
r2^(-1),f1^-1,r3^(-1),f1^-1,r2^(-1),f1^-1,r2,f1^-1,r2^(-1),f1^-1,
r2,r3^(-1),f1^-1,r2^(-1),f1^-1,r3^(-1),f1^-1,r2,f1^-1,
r3,f1^-1,r2^(-1),f1^-1,r3,f1^-1,r2,f1^-1,r2^(-1),
f1^-1,r2,f1^-1,r2^(-2),r3,f1^-1,r2,f1^-1,
r2^(-2),f1^-1,r2^(-1),f1^-1,f3,f4^-1];

```

Another 2-cycle on polar facets:

The following move acts as the permutation (4 ,28) on the masterball:

```

NS_swap:=[r4,r1,f1,r1,f1,r4,f1,r1^(-1),f1,r4^(-2),
f1,r1,f1,r4,f1,r1^(-1),f1,r4^4,f1,
r1,f1,r4,f1,r1^(-1),f1,r4^(-2),f1,r1,f1,
r1^(-1),r4^2,r1,f1,r1^(-1),f1,r4^2,f1,r1,
f1,r4^(-1),f1,r1^(-1),f1,r4^(-1),f1,r1,f1,
r4,f1,r1^(-1),f1,r1,r4^(-2),f1,r1^(-1),f1,
r4^(-1),f1,r1,f1,r1^(-1),r4,f1,r1,f1,
r1^(-1),f1,r1,f1,r1^(-1),r4^2,r1,f1,
r1^(-1),f1,r1,f1,r1^(-1),f1,r1,r4^(-1),
f1,r1^(-1),f1,r4,f1,r1,f1,r1^(-1),r4^2,r1^(-1)];

```

3.4 The solution using GAP

This idea seems to work in principal but in practice is can lead to solutions which are too long to be practical.

To solve the rainbow masterball, assume that you have represented the initial position as an 4×8 matrix $A0$.

You need to determine the element of the masterball group $G \subset S_{32}$ corresponding to this position. Call this element $L0$, written as a list of lists in MAPLE's disjoint cycle notation.

The following question naturally arises: Can we find (or at least get MAPLE or GAP to find) an expression for $L0$ as a product of the generators $r1, \dots, f8$ of G ? This is a special instance of the "word problem" in group theory.

I shall now assume that you have the GAP 3.4 package (GAP is a freeware package produced by Lehrstuhl D für Mathematik, RWTH Aachen - you may try to get it by anonymous ftping [ftp.math.rwth-aachen.de](ftp://ftp.math.rwth-aachen.de)) and the GAP share package AbStab.g (loaded as abstab.g in the directory `c:\gap\gap3r4p3\lib\abstab.g` if you have a dos machine, otherwise you will need to modify the rainbow.g file included in an appendix below).

After starting gap, log your session into some text file, rainbow.log say, by typing

```
LogTo("rainbow.log");
```

Now load the rainbow.g file by typing

```
Read("c:/gap/rainbow/rainbow.g");
```

assuming that your path to the file is `c:\gap\rainbow` (otherwise, modify the path accordingly). Enter the permutation element describing the MasterBall position into GAP. Call it w . (The GAP notation for a permutation is not the same as the MAPLE notation but they are similar. There are some MAPLE procedures included in rainbow.txt to help you describe this permutation but they, unfortunately, do not always return disjoint cycles.) The command

```
w in G;
```

will return "true" if w is an element of the masterball group G . The AbStab command

```
soln:=FactorPermGroupElement(G, w );
```

will return the solution of the masterball as a word in the generators $g1, g2, \dots$ choosen by AbStab. This can take some time and yield a fairly long expression, depending on w . The AbStab command

```
soln_short:=Shrink(G, w );
```

will try to produce a shorter solution of the masterball as a word in the generators $g1, g2, \dots$ choosen by AbStab.

These generators are a subset of the generators $r1, \dots, f8$. You need to find out which is which. For example, to find which (if any) generator $r1$ is, type

```
FactorPermGroupElement(G, r1 );
```

For example, you might find (as I did in one session)

```
g1=r4, g2=r3, g3=r2, g4=f5, g5=f4, g6=f3, g7=f2, g8=r1.
```

In any case, you need to know these g_i 's. Now the permutation `soln` is in GAP notation. By exiting GAP and loading `rainbow.log` into an ASCII text editor you can modify `soln` into MAPLE notation (replace all the g_i 's by the appropriate r_i or f_i , replace '*' by ',', replace $\hat{-}1$ by $\hat{(-1)}$ for example, and save the resulting expression as a list by putting a [in front and a]; in back). This can now be pasted into MAPLE and the masterball solved using the `rb` command as discussed in the previous subsection.

Remark 1 *What is remarkable is that, at least in principle, we have an “expert system” (one which can solve the masterball) with no built-in “knowledge” of the masterball other than its group structure.*

4 Appendix: Code

We include the contents of the file `rainbow.txt` (MAPLE), `rainbow.g` (GAP).

4.1 Maple code: `rainbow.txt`

```
r1 := [[1, 2, 3, 4, 5, 6, 7, 8]];
r2 := [[9, 10, 11, 12, 13, 14, 15, 16]];
r3 := [[17, 18, 19, 20, 21, 22, 23, 24]];
r4 := [[25, 26, 27, 28, 29, 30, 31, 32]];
f5 := [[1,28],[2,27],[3,26],[4,25],[9,20],[10,19],[11,18],[12,17]];
f4 := [[2,29],[3,28],[4,27],[5,26],[10,21],[11,22],[12,23],[13,24]];
f3 := [[3,30],[4,29],[5,28],[6,27],[11,22],[12,21],[13,20],[14,19]];
f2 := [[4,31],[5,30],[6,29],[7,28],[12,23],[13,22],[14,21],[15,20]];
f1 := [[5,32],[6,31],[7,30],[8,29],[13,24],[14,23],[15,22],[16,21]];
f8 := [[6,25],[7,32],[8,31],[1,30],[14,17],[15,24],[16,23],[9,22]];
f7 := [[7,26],[8,25],[1,32],[2,31],[15,18],[16,17],[9,24],[10,23]];
f6 := [[8,27],[1,26],[2,25],[3,32],[16,19],[9,18],[10,17],[11,24]];

order_of_group := 437763136697395052544000000;
#divide this by 2^20 to get the number of distinct positions
```

```

perm_ball
:= permgroup(32,{
r1=[[1, 2, 3, 4, 5, 6, 7, 8]],
r2=[[9, 10, 11, 12, 13, 14, 15, 16]],
r3=[[17, 18, 19, 20, 21, 22, 23, 24]],
r4=[[25, 26, 27, 28, 29, 30, 31,32]],
f5=[[1,28],[2,27],[3,26],[4,25],[9,20],[10,19],[11,18],[12,17]],
f4=[[2,29],[3,28],[4,27],[5,26],[10,21],[11,22],[12,23],[13,24]],
f3=[[3,30],[4,29],[5,28],[6,27],[11,22],[12,21],[13,20],[14,19]],
f2=[[4,31],[5,30],[6,29],[7,28],[12,23],[13,22],[14,21],[15,20]],
f1=[[5,32],[6,31],[7,30],[8,29],[13,24],[14,23],[15,22],[16,21]],
f8=[[6,25],[7,32],[8,31],[1,30],[14,17],[15,24],[16,23],[9,22]],
f7=[[7,26],[8,25],[1,32],[2,31],[15,18],[16,17],[9,24],[10,23]],
f6=[[8,27],[1,26],[2,25],[3,32],[16,19],[9,18],[10,17],[11,24]]));
#this is "the group" of the masterball - it does not act freely

flip_poles := [[1, 25], [2, 26], [3, 27], [4, 28], [5, 29],
[6, 30], [7, 31], [8, 32]];
#this group element flips the facelets from the north pole
#to the south pole, leaving the middle facelets intact
#It *does* belong to perm_group (and obviously acts trivially
#on the solved masterball). It also belongs to the derived subgroup.

flip_wide_bands := [[9, 17], [10, 18], [11, 19], [12, 20],
[13, 21], [14, 22], [15, 23], [16, 24]];
#this group element flips the facelets from the upper middle band
#to the lower middle band, leaving the poles intact
#It *does* belong to perm_group (and obviously acts trivially
#on the masterball). It also belongs to the derived subgroup.

long2cycle :=
[f1,f3,r1^(-1),f3,f1,r4^(-1),r4^(-1),f1,f3,r1,f3,f1,r4^(-1),f1,
f3,r1^(-1),f3,f1,r4^(-1),f1,f3,r1^(-1),f3,f1,r4^(-1),f1,
f3,r1,r1,f3,f1,r4^(-1),f1,f3,r1^(-1),f3,f1,r4,f1,
f3,r1,f3,f1,r4^(-1),r4^(-1),f1,f3,r1^(-1),f3,f1,r4^(-1),f1,
f3,r1^(-1),f3,f1,r4^(-1),f1,f3,r1,f3,f1,r4^(-1),r4^(-1),r4^(-1),
f1,f3,r1^(-1),f3,f1,r4^(-1),f1,f3,r1^(-1),f3,f1,
```

```

r4^(-1),f1,f3,r1,f3,f1,r4^(-1),f1,f3,r1,f3,f1,r4,
f1,f3,r1^(-1),r1^(-1),f3,f1,r4,f1,f3,r1,f3,f1,r4,f1,
f3,r1,f3,f1,r4,f1,f3,r1^(-1),f3,f1,r4,f1,
f3,r1,f3,f1,r4,f1,f3,r1,f3,f1,r4,f1,
f3,r1^(-1),f3,f1,r4,r4,f1,f3,r1,f3,f1,
r4^(-1),r4^(-1),r4^(-1),f1,f3,r1^(-1),f3,f1,r4^(-1),f1,f3,r1^(-1),
f3,f1,r4^(-1),f1,f3,r1,r1,f3,f1,r4^(-1),r4^(-1),r4^(-1),
f1,f3,r1^(-1),f3,f1,r4^(-1),f1,f3,r1^(-1),f3,
f1,r4^(-1),f1,f3,r1,f3,f1,r4^(-1),f1,f3,r1,
f3,f1,r4,f1,f3,r1^(-1),r1^(-1),f3,f1,r4,f1,f3,
r1,f3,f1,r4,f1,f3,r1,f3,f1,r4,f1,f3,
r1^(-1),f3,f1,r4,r4,f1,f3,r1^(-1),f3,f1,
r4,f1,f3,r1,f3,f1,r4,f1,f3,r1,f3,f1,
r4,r4,r4,f1,f3,r1^(-1),r1^(-1),f3,f1,r4,f1,f3,r1,f3,
f1,r4,f1,f3,r1,f3,f1,r4,r4,f1,f3,r1^(-1),
f3,f1,r4^(-1),f1,f3,r1,f3,f1,r4,f1,f3,
r1^(-1),r1^(-1),f3,f1,r4,f1,f3,r1,f3,f1,r4,f1,
f3,r1,f3,f1,r4,f1,f3,r1^(-1),f3,f1,
r4,r4,f1,f3,r1,f3,f1,r4^(-1),r4^(-1),f1,f3,r1^(-1),
f3,f1,r4,f1,f3,r1,f3,f1,r4^(-1),r4^(-1),f1,
f3,r1^(-1),f3,f1,r4^(-1),f1,f3,r1^(-1),f3,f1,
r4^(-1),f1,f3,r1,r1,f3,f1,r4^(-1),r4^(-1),r4^(-1),f1,f3,r1^(-1),
f3,f1,r4^(-1),f1,f3,r1^(-1),f3,f1,r4^(-1),
f1,f3,r1,r1,f3,f1,r4^(-1),f1,f3,r1^(-1),f3,
f1,r4^(-1),f1,f3,r1,f3,f1,r4,r4];
#this manuever evaluates to the 2-cycle [[1,2]]

```

```

der_perm_ball := permgroup(32,{
[[1, 26, 30], [2, 6, 25], [3, 7, 32], [8, 27, 31],
[9, 18, 22], [10, 14, 17], [11, 15, 24],
[16, 19, 23]],
[],
[[1, 5, 30, 7, 32], [6, 31, 8, 25, 29], [9, 13, 22, 15, 24],
[14, 23, 16, 17, 21]],
[[9, 22, 23, 24, 17, 18, 14, 15, 16]],
[[1, 30, 5], [4, 8, 31], [6, 29, 25], [7, 28, 32], [9, 22, 13],
[12, 16, 23], [14, 21, 17], [15, 20, 24]],
```

```

[[1, 30, 31, 32, 25, 26, 6, 7, 8]]});
#this is the derived group of perm_ball

order_of_der_perm_ball := 109440784174348763136000000;
#this is 1/4th order_of_group

der_2_perm_ball := permgroup(32,{
[], [[1, 6, 7, 8, 30, 2, 3], [25, 31, 32, 26, 27]],
[[1, 26], [2, 25], [3, 32], [4, 31], [5, 30], [6, 29],
[7, 28], [8, 27], [9, 18], [10, 17], [11, 24], [12, 23],
[13, 22], [14, 21], [15, 20], [16, 19]],
[[1, 30, 7, 32, 5, 26, 3], [2, 27, 25, 6, 31, 8, 29],
[9, 22, 15, 24, 13, 18, 11], [10, 19, 17, 14, 23, 16, 21]],
[[9, 14, 15, 16, 22, 10, 11], [17, 23, 24, 18, 19]]]);
#this is the derived group of the derived group of perm_ball

order_of_der_2_perm_ball := 109440784174348763136000000;
#its order is the same as der_perm_ball

mod8 := proc (j)
local k, l;
k := `mod`(j,8); if k = 0 then l := 8 else l := k
fi; RETURN(l) end;

mod84_row := proc (i, j, n)
local temp, tempo;
if i = 1 then temp := 4 else if
i = 2 then temp := 3 else if i = 3 then
temp := 2 else if i = 4 then temp := 1
fi fi fi fi;
if j = n or j = mod8(n+1) or j = mod8(n+2) or j = mod8(n+3) then
tempo := temp else temp := i fi;
RETURN(temp)
end;

mod84_col := proc (i, j, n)
local temp;
if j = n then temp := mod8(n+3) else

```

```

if j = mod8(n+1) then temp := mod8(n+2) else if j = mod8(n+2) then
temp := mod8(n+1) else if j = mod8(n+3) then temp := n else
temp := j
fi fi fi fi;
RETURN(temp)
end;

move_long := proc (A, i)
local j, k, AA;
AA:=array(1..4,1..8);
#print('MOVE_LONG');
for k to 8 do
for j to 4 do
AA[j,k] := A[mod84_row(j,k,i),mod84_col(j,k,i)]
od
od;
#print(AA,'AAMATRIX_IN_MOVE_LONG');
RETURN(AA)
end;

color1 := red;
color2 := blue;
color3 := coral;
color4 := plum;
color5 := green;
color6 := yellow;
color7 := cyan;
color8 := COLOR(RGB,.5607,.7372,.5607);

master_squares := proc ()
local i;
for i from 0 to 3 do
point(A1.i,0,i), point(B1.i,1,i), point(C1.i,1,1+i), point(D1.i,0,1+i);
square(Sq.1.i,[A1.i, B1.i, C1.i, D1.i]);
point(A2.i,1,i), point(B2.i,2,i), point(C2.i,2,1+i), point(D2.i,1,1+i);
square(Sq.2.i,[A2.i, B2.i, C2.i, D2.i]);
point(A3.i,2,i), point(B3.i,3,i), point(C3.i,3,1+i), point(D3.i,2,1+i);
square(Sq.3.i,[A3.i, B3.i, C3.i, D3.i]);

```

```

point(A4.i,3,i), point(B4.i,4,i), point(C4.i,4,1+i), point(D4.i,3,1+i);
square(Sq.4.i,[A4.i, B4.i, C4.i, D4.i]);
point(A5.i,4,i), point(B5.i,5,i), point(C5.i,5,1+i), point(D5.i,4,1+i);
square(Sq.5.i,[A5.i, B5.i, C5.i, D5.i]);
point(A6.i,5,i), point(B6.i,6,i), point(C6.i,6,1+i), point(D6.i,5,1+i);
square(Sq.6.i,[A6.i, B6.i, C6.i, D6.i]);
point(A7.i,6,i), point(B7.i,7,i), point(C7.i,7,1+i), point(D7.i,6,1+i);
square(Sq.7.i,[A7.i, B7.i, C7.i, D7.i]);
point(A8.i,7,i), point(B8.i,8,i), point(C8.i,8,1+i), point(D8.i,7,1+i);
square(Sq.8.i,[A8.i, B8.i, C8.i, D8.i]) od
end;

square_draw := proc (A)
local L, i, j;
master_squares(); L := []; for j to 8
do for i from 0 to 3 do
L := [op(L), Sq.j.i(color = color.(A[i+1,j]))] od od;
draw(L,filled = true,printtext = false,scaling = constrained,axes = none)
end;

sphere_draw := proc (A::array)
local i, j, p1, p2, p3, p4, B11, B12, B13, B14,
B15, B16, B17, B18, B21, B22, B23, B24, B25, B26,
B27, B28, B31, B32, B33, B34, B35, B36, B37, B38,
B41, B42, B43, B44, B45, B46, B47, B48, L;
for i to 8 do
p1.i := [8*cos(1/4*i*Pi), 8*sin(1/4*i*Pi), 10/sqrt(2)] od;
for i to 8 do p2.i:= [10*cos(1/4*i*Pi), 10*sin(1/4*i*Pi), 0] od;
for i to 8 do p4.i := [8*cos(1/4*i*Pi), 8*sin(1/4*i*Pi), -10/sqrt(2)] od;
B11:=polygonplot3d([p11,p12,[0,0,10]],style=patch,color=color.(A[1,1]));
B12:=polygonplot3d([p12,p13,[0,0,10]],style=patch,color=color.(A[1,2]));
B13:=polygonplot3d([p13,p14,[0,0,10]],style=patch,color=color.(A[1,3]));
B14:=polygonplot3d([p14,p15,[0,0,10]],style=patch,color=color.(A[1,4]));
B15:=polygonplot3d([p15,p16,[0,0,10]],style=patch,color=color.(A[1,5]));
B16:=polygonplot3d([p16,p17,[0,0,10]],style=patch,color=color.(A[1,6]));
B17 := polygonplot3d([p17,p18,[0,0,10]],style=patch,color=color.(A[1,7]));
B18:=polygonplot3d([p18, p11, [0, 0, 10]],style = patch,color =
color.(A[1,8]));

```

```

B21 := polygonplot3d([p11, p12, p22, p21],style = patch,color
= color.(A[2,1]));
B22 := polygonplot3d([p12, p13, p23, p22],style = patch,
color = color.(A[2,2]));
B23 := polygonplot3d([p13, p14, p24, p23],style = patch,
color = color.(A[2,3]));
B24 := polygonplot3d([p14, p15, p25, p24],style= patch,
color = color.(A[2,4]));
B25 := polygonplot3d([p15, p16, p26, p25],style = patch,
color = color.(A[2,5]));
B26 := polygonplot3d([p16, p17, p27, p26],style = patch,
color = color.(A[2,6]));
B27 := polygonplot3d([p17, p18, p28, p27],style = patch,
color = color.(A[2,7]));
B28 := polygonplot3d([p18, p11, p21, p28],style = patch,
color = color.(A[2,8]));
B31 := polygonplot3d([p41, p42, p22, p21],style = patch,
color = color.(A[3,1]));
B32 := polygonplot3d([p42, p43, p23, p22],style = patch,
color = color.(A[3,2]));
B33:= polygonplot3d([p43, p44, p24, p23],style = patch,
color = color.(A[3,3]));
B34 := polygonplot3d([p44, p45, p25, p24],style = patch,
color = color.(A[3,4]));
B35 := polygonplot3d([p45, p46, p26, p25],style = patch,
color = color.(A[3,5]));
B36 := polygonplot3d([p46, p47, p27, p26],style = patch,
color = color.(A[3,6]));
B37 := polygonplot3d([p47, p48, p28, p27],style = patch,
color = color.(A[3,7]));
B38 := polygonplot3d([p48, p41, p21, p28],style = patch,
color = color.(A[3,8]));
B41 := polygonplot3d([p41, p42, [0, 0, -10]],style = patch,
color = color.(A[4,1]));
B42 := polygonplot3d([p42, p43, [0, 0, -10]],style =patch,
color = color.(A[4,2]));
B43 := polygonplot3d([p43, p44, [0, 0, -10]],style = patch,
color = color.(A[4,3]));

```

```

B44 := polygonplot3d([p44, p45, [0, 0, -10]], style = patch,
color = color.(A[4,4]));
B45 := polygonplot3d([p45, p46, [0, 0, -10]], style = patch,
color = color.(A[4,5]));
B46 := polygonplot3d([p46,p47, [0, 0, -10]], style = patch,
color = color.(A[4,6]));
B47 := polygonplot3d([p47, p48, [0, 0, -10]], style = patch,
color = color.(A[4,7]));
B48 := polygonplot3d([p48, p41, [0, 0, -10]], style = patch,
color = color.(A[4,8]));
L:= [B21, B22, B23, B24, B25, B26, B27, B28,
B11, B12, B13, B14, B15, B16, B17,B18,
B31, B32, B33, B34, B35, B36, B37, B38,
B41, B42, B43, B44, B45, B46, B47, B48];
RETURN(L)
end;

move_lat_pow := proc (A::array,i,pow)
local j, k, AA;
#print('MOVE_LAT_POW',i,pow);
AA:=array(1..4,1..8);
for k to 8 do
for j to 4 do if j = i then AA[j,k] := A[j,mod8(k+pow)] 
else AA[j,k] := A[j,k] fi od od;
#print(AA,'AAMATRIX_IN_MOVE_LAT_POW');
RETURN(AA)
end;

move_lat_power := proc (A::array,i,n)
local nn, j, BB;
#print('MOVE_LAT_POWER');
BB:=array(1..4,1..8);
if n=0 then RETURN(A); fi;
nn := mod8(n);
###print('POWER',nn);
BB := move_lat_pow(A,i,nn);
#print(BB,'BBMATRIX_IN_MOVE_LAT_POWER');
RETURN(BB)

```

```

end;

power_move:= proc(g,A::array)
#g should be a power of a generator or 1
#A should be a 4x8 matrix
local i,j,col,row,power,gen,
AA, generators1, generators2, generators;
AA:=array(1..4,1..8);
generators1 := {r1, r4, r3, r2};
generators2 := {f1, f2, f3, f4, f5, f6, f7, f8};
generators := 'union'(generators1,generators2);
gen:= op(1,g);
power := op(2,g);
for i from 1 to 4 do
if r.i=gen then
row := i;
AA:=move_lat_pow(A,row,'mod'(power,8));
fi;
od;
for j to 8 do if gen = f.j then
col := j;
if 'mod'(power,2)<>0 then
AA := move_long(A,col) else AA:=A; fi;
fi
od;
if g=1 then AA:=A; fi;
RETURN(AA);
end;

move := proc (L, A::array)
#L should be a list or 1
#A should be a 4x8 matrix
local i,j,k, col,row,power,gen,
DD,EE, length_L,generators1, generators2, generators;
DD:=array(1..4,1..8);
EE:=array(1..4,1..8);
generators1 := {r1, r4, r3, r2};
generators2 := {f1, f2, f3, f4, f5, f6, f7, f8};

```

```

generators := 'union'(generators1,generators2);
if type(L,'^') then
EE:=power_move(L,A);
fi;
if member(L,generators1) then
for j from 1 to 4 do
if L=r.j then
RETURN(move_lat_pow(A,j,1));
fi;
od; ##### for j
fi;
if member(L,generators2) then
for k from 1 to 8 do
if L=f.k then
RETURN(move_long(A,k));
fi;
od; ##### for k
fi;
if L=1 then RETURN(A); fi;
if type(L,list) then
length_L := nops(L);
DD:=A; ##### initializing DD
for i to length_L do
EE:=eval(DD,i); ##### new, was EE:=DD;
if type(op(i,L),'^') then
DD:=move(op(i,L),EE); ##### new line
fi; ##### if type(op(i,L),'^') then
if member(op(i,L),generators1) then
for j from 1 to 4 do
if op(i,L)=r.j then
DD := move_lat_pow(EE,j,1);
fi;
od;
fi;
if member(op(i,L),generators2) then
for k from 1 to 8 do
if op(i,L)=f.k then
DD := move_long(EE,k);

```

```

fi; ##### if op(i,L)=f.k then
od; ##### for k
fi; ### if member(op(i,L),generators2) then
EE:=eval(DD,1); ##### new, was EE:=DD;
od; ##### for i
fi;
RETURN(EE)
end;

color_count := proc (A::array)
local count, rows, cols, i, j, count_okay, k;
for i to 8 do count[i] := 0 od;
count_okay := 'true'; rows := rowdim(A); cols:= coldim(A);
if rows <> 4 or cols <> 8 then
ERROR('The input matrix must be 4x8',rows,cols) fi;
for i to 4 do for j to 8 do
if not (type(A[i,j],integer) and 0 < A[i,j] and A[i,j] < 9) then
ERROR('The input matrix must be integral with entries in [1,8]',A[i,j]) fi
od od;
for i to 4 do for j to 8 do for k to 8 do if A[i,j] =
k then count[k] := 1+count[k] fi od od od;
for i to 8 do if count[i] <> 4 then
count_okay := 'false' fi od;
RETURN(count_okay)
end;

matrix_to_perm1 := proc (A::array)
#this procedure will not work properly unless the
#cycles are disjoint
local L, i, j, k;
if color_count(A) <> true then
ERROR('Must have 8 colors each occurring 4 times') fi;
L := []; for i to 4 do for j to 8 do
if i = 1 and A[i,j] <> j and not member([j, A[i,j]],L)
then L := [op(L), [A[i,j], j]] fi;
if i =2 and A[i,j] <> j and not member([j+8, A[i,j]+8],L)
then L := [op(L), [A[i,j]+8, j+8]] fi;
if i = 3 and A[i,j] <> j and not member([j+16, A[i,j]+16],L) then

```

```

L := [op(L), [A[i,j]+16, j+16]] fi;
if i = 4 and A[i,j] <> j and not member([j+24, A[i,j]+24],L)
then L := [op(L), [A[i,j]+24, j+24]] fi od od;
RETURN(L)
end;

color_position := proc (i, k, A)
local L, j; L := [] ; for j to 8 do if A[i,j]
= k then L := [op(L), j] fi od;
RETURN(L)
end;

double_colors := proc (i, A)
local j, k, L; L := [] ;
for k to 8 do if 1 < nops(color_position(i,k,A))
then L := [op(L), k] fi od;
RETURN(L)
end;

undouble_colors := proc (MM::array)
local i, j, k, A_mat, d, col1, col2, col3,
col4, dd1, dd4, dd2, dd3, L, num_double_colors1,
num_double_colors2;
A_mat := matrix(4,8);
for i to 4 do for j to 8 do A_mat[i,j] := MM[i,j] od od;
L := [];
num_double_colors1 := nops(double_colors(1,MM));
num_double_colors2 := nops(double_colors(2,MM));
if nops(double_colors(1,MM)) <> nops(double_colors(4,MM))
then
ERROR('Colors are not possible for a rainbow matrix')
fi;
if nops(double_colors(2,MM)) <> nops(double_colors(3,MM)) then
ERROR('Colors are not possible for a rainbow matrix')
fi;
if 0 < num_double_colors1 then for d to nops(double_colors(1,MM)) do
dd1:= op(d,double_colors(1,MM));
dd4 := op(d,double_colors(4,MM));

```

```

col1 := op(1,color_position(1,dd1,MM));
col4 := op(1,color_position(4,dd4,MM));
L := [op(L), [col1, 24+col4]]; A_mat[1,col1] := MM[4,col4];
A_mat[4,col4] := MM[1,col1] od fi;
if 0 < num_double_colors2 then for d to nops(double_colors(2,MM)) do
dd2 := op(d,double_colors(2,MM));
dd3 := op(d,double_colors(3,MM));
col2 := op(1,color_position(2,dd2,MM));
col3 := op(1,color_position(3,dd3,MM));
L := [op(L), [8+col2, 16+col3]]; A_mat[2,col2] := MM[3,col3];
A_mat[3,col3] := MM[2,col2] od fi;
RETURN(A_mat)
end;

undouble_colors_perm := proc (MM::array)
local i, j, k, d, col1, col2, col3,
col4, dd1, dd4, dd2, dd3, L, num_double_colors1,
num_double_colors2;
L := [];
num_double_colors1 := nops(double_colors(1,MM));
num_double_colors2 := nops(double_colors(2,MM));
if nops(double_colors(1,MM)) <> nops(double_colors(4,MM)) then
ERROR('Colors are not possible for a rainbow matrix')
fi;
if nops(double_colors(2,MM)) <> nops(double_colors(3,MM)) then
ERROR('Colors are not possible for a rainbow matrix')
fi;
if 0 < num_double_colors1 then for d to nops(double_colors(1,MM)) do
dd1:= op(d,double_colors(1,MM)); dd4 := op(d,double_colors(4,MM));
col1 := op(1,color_position(1,dd1,MM));
col4 := op(1,color_position(4,dd4,MM));
L := [op(L), [col1, 24+col4]] od fi; if 0 < num_double_colors2 then
for d to nops(double_colors(2,MM)) do
dd2 := op(d,double_colors(2,MM)); dd3 := op(d,double_colors(3,MM));
col2 := op(1,color_position(2,dd2,MM));
col3 := op(1,color_position(3,dd3,MM));
L := [op(L), [8+col2, 16+col3]] od fi;
RETURN(L)

```

```

end;

matrix_to_perm := proc (MM::array)
local L, rows, cols;
rows := rowdim(MM);
cols := coldim(MM);
if rows <> 4 or cols <> 8 then
ERROR('The input matrix must be 4x8',rows,cols) fi;
L := [op(undouble_colors_perm(MM)),
op(matrix_to_perm1(undouble_colors(MM)))] ;
RETURN(L)
end;

rb := proc (L, A) local AA, LL;
AA := move(L,A); LL := sphere_draw(AA);
display3d(LL) end;

```

4.2 GAP code: rainbow.g

```

r1 := (1, 2, 3, 4, 5, 6, 7, 8);
r2 := (9, 10, 11, 12, 13, 14, 15, 16);
r3 := (17, 18, 19, 20, 21, 22, 23, 24);
r4 := (25, 26, 27, 28, 29, 30, 31, 32);
f5 := (1, 28)(2, 27)(3, 26)(4, 25)(9, 20)(10, 19)(11, 18)(12, 17);
f4 := (2, 29)(3, 28)(4, 27)(5, 26)(10, 21)(11, 22)(12, 23)(13, 24);
f3 := (3, 30)(4, 29)(5, 28)(6, 27)(11, 22)(12, 21)(13, 20)(14, 19);
f2 := (4, 31)(5, 30)(6, 29)(7, 28)(12, 23)(13, 22)(14, 21)(15, 20);
f1 := (5, 32)(6, 31)(7, 30)(8, 29)(13, 24)(14, 23)(15, 22)(16, 21);
f8 := (6, 25)(7, 32)(8, 31)(1, 30)(14, 17)(15, 24)(16, 23)(9, 22);
f7 := (7, 26)(8, 25)(1, 32)(2, 31)(15, 18)(16, 17)(9, 24)(10, 23);
f6 := (8, 27)(1, 26)(2, 25)(3, 32)(16, 19)(9, 18)(10, 17)(11, 24);

order_of_group := 437763136697395052544000000;

Read("c:/gap/gap3r4p3/lib/abstab.g");

G:=Group(r1,r2,r3,r4,f1,f2,f3,f4,f5,f6,f7,f8);

```

5 Other sources of information

To obtain a MasterBall, you may want to contact
Puzzletts, Inc.

24843 144th Place S.E.
Kent, WA 98042-3423
(206) 630-1432
<http://www.puzzletts.com/>

Some new cubes are sold instructions and hints. Otherwise, I know of no documentation. (However, I am told by Puzzlett owner Mike Green that documentation exists for the rainbow masterball.)

Some more information on the rainbow package can be found on my www page <http://www.nadn.navy.mil/MathDept/wdj/myhome.html>

References

- [B] C. Bandelow, Inside Rubik's cube and beyond, Birkhäuser Boston, 1980